Optimal Energy Allocation Policy for Wireless Networks in the Sky

Dinh Thai Hoang[1], Dusit Niyato[1] and Nguyen Tai Hung[2]
[1] School of Computer Engineering, Nanyang Technological University (NTU), Singapore
[2] School of Electronics and Telecommunications, Hanoi University of Science and Technology, Vietnam

***Abstract*—Google's Project Loon [1] was launched in 2013 with the aim of providing Internet access to rural and remote areas. In the Loon network, balloons travel around the Earth and bring access points to the users who cannot connect directly to the global wired Internet. The signals from the users will be transmitted through the balloon network to the base stations connected to the Internet service provider (ISP) on Earth. The process of transmitting and receiving data consume a certain amount of energy from the balloon, while the energy on balloons cannot be supplied by stable power source or by replacing batteries frequently. Instead, the balloons can harvest energy from natural energy sources, e.g., solar energy, or from radio frequency energy by equipping with appropriate circuits. However, such kinds of energy sources are often dynamic and thus how to use this energy efficiently is the main goal of this paper. In this paper, we study the optimal energy allocation problem for the balloons such that network performance is optimized and the revenue for service providers is maximized. We first formulate the stochastic optimization problem as a Markov decision process and then apply a learning algorithm based on simulation-based method to obtain optimal policies for the balloons. Numerical results obtained by extensive simulations clearly show the efficiency and convergence of the proposed learning algorithm.**

Keywords- Internet in the sky, Google Loon Project, Markov decision process.

I. Introduction

After more than 40 years of development Internet has created a revolution in communication for humans because it allows people to access and exchange information efficiently. Although Internet is highly accessible, approximately 60-70% of people worldwide do not have the Internet reported by International Telecommunications Union [2] in June 2013. This stems from a fact that many areas such as Africa, Asia, and Pacific, cannot offer Internet connections due to geographical and infrastructure issues. Therefore, the idea of providing Internet connections via wireless networks has become more and more popular.

In wireless Internet, mobile users can connect to the Internet service provider (ISP) through base stations or access points. However, deployment of base stations for every location on the Earth seems to be impossible, e.g., oceans and mountains. Therefore, the idea of providing Internet from the sky was introduced. The early version is based on the satellites, which suffers from high cost and long transmission delay [3]. As a result, the cheaper and faster alternative, i.e., Google Loon project [1], was proposed. In Loon project, access points will be placed on balloons flying at an altitude of about 20 km which is safe from bad weather and flights. The balloons will travel around the Earth and form a network of access points for Internet users in remote places. When receiving data from the user, the balloon will find the shortest route to transfer data to the nearest base station on the ground, which will be forwarded to an ISP.

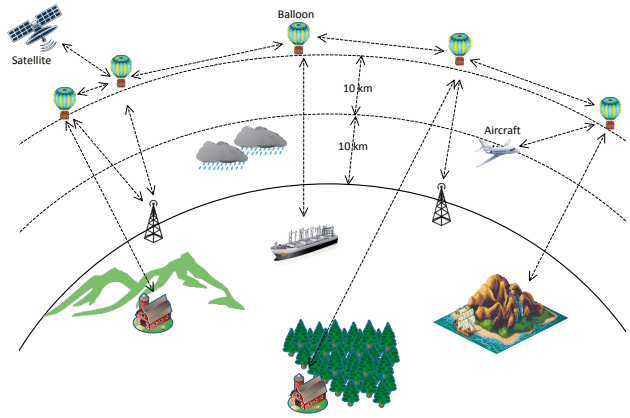


Fig. 1: The model of wireless network in the sky.

Data transmission and reception by the access points on the balloon consume energy, which requires continuous supply to sustain the operations. Only viable energy sources for the balloon are through energy harvesting such as solar energy or radio frequency (RF). The harvested energy will be stored in the energy storage of the access points and it will be used for data transmission and reception. However, batteries equipped on balloons are often limited by size and energy harvested from solar or RF is random. Moreover, the balloons may have to serve data transfer requests from different types of users, e.g., from other balloons, on the ground, satellites, or aircrafts, with different quality of service (QoS) requirements. Therefore, energy management for the balloons is an important issue.

In this paper, we aim to find an optimal admission control policy for the access points deployed on the balloons. The goal is to ensure high energy efficiency while maximizing profit of service providers. We formulate a Markov decision process (MDP) for the energy allocation optimization problem. To obtain the optimal policy, we apply a learning algorithm based on the policy gradient method and simulation-based method. The proposed learning algorithm not only avoids the curse of dimensionality problem caused by the explosion of state

and action spaces, but also eliminates the need for complete knowledge about the model, which may not be possible to have from an unpredictable environment. Numerical results show the convergence as well as the efficiency of the proposed learning algorithm.

II. System Model

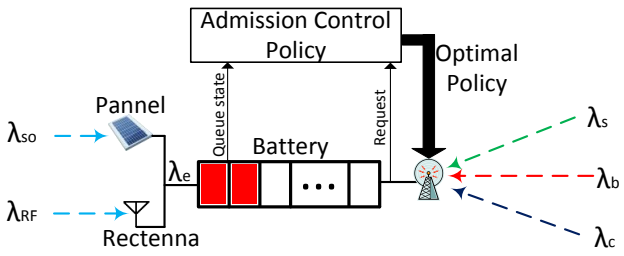


Fig. 2: The system model.

We consider energy management problem for a balloon with an access point as shown in Fig. 2. Specifically, the access point is to receive and transmit data from users. The access point has a battery to store energy harvested from solar and RF. When a request is sent to the access point, it will check the amount of energy remaining in the battery and apply an admission control policy by deciding whether the request will be accepted or rejected. If the request is accepted, a certain amount of energy from the battery will be used to receive data and transmit data to the next hop. Different requests have different QoS requirements. Therefore, we divide requests into three classes, i.e., requests from other balloons (i.e., class-1), requests from users on the ground (class-2), and requests from satellites or aircrafts (class-3). The arrival processes of requests from class-1, class-2, and class-3 are assumed to follow the Poisson distribution with mean rates λ_b, λ_c, and λ_s, respectively. When a request is accepted, the balloon will receive an immediate reward (i.e., revenue). The immediate revenues for accepting requests from class-1, class-2, and class-3 are r_b, r_c, and r_s, respectively.

In the Loon network, balloons are assumed to be equipped with solar panels [4] for harvesting energy from sunlight. Additionally, we assume that the balloons can be equipped with a rectenna [5] to harvest energy from RF waves. We assume that the energy arrival from both solar panel and RF rectenna follows the Poisson distribution with mean rates λ_e, and the successful energy harvesting probability is p_e^{su}. The maximum capacity of the battery is E. At each time epoch, when the access point receives a request, it will consult with the admission control policy. The admission control policy will determine a decision to accept or reject the request based on the request' class and the current energy level in the battery. In this paper, we are interested in maximizing the profit for the service provider in the terms of average reward for the balloon.

III. Problem Formulation

In this Section, we will formulate the optimization problem as Markov decision process (MDP) and study a learning algorithm to obtain the optimal policy for balloons.

A. MDP Framework

An MDP is defined by a tuple of $< \mathcal{S}, \mathcal{A}, P, R >$ where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, P is a transition probability function, and R is a reward function. The state space $\mathcal{S}$ of our admission control for the access point on a balloon is defined as follows:

$$\mathcal{S} \triangleq \Big\{ \mathbf{s} = (\mathbf{e}, \mathbf{x}) : \mathbf{e} \in \mathcal{E}; \mathbf{x} \in \mathcal{X} \Big\}, \tag{1}$$

where $\mathcal{E} = \{0, 1, \ldots, E\}$ is the energy state space whose elements represent energy levels in the battery. $\mathcal{X} = \{x_b, x_c, x_s, x_e\}$ is a set of events in which x_b, x_c, and x_s are the events when a request is from another access point, a user, and a satellite, respectively. x_e is an event for energy arrival.

When the system is at state $\mathbf{e}$, if an event $\mathbf{x}$ happens, an accept/reject decision must be made. Thus, we can define the action space as follows:

$$\mathcal{A} \triangleq \Big\{ \mathbf{a} : \mathbf{a} \in \{0, 1\} \Big\}, \tag{2}$$

where

$$\mathbf{a} = \begin{cases} 1, & \text{if the arriving request is accepted} \\ 0, & \text{if the arriving request is rejected} \end{cases}$$

To derive the transition probability function $P(\mathbf{s}, \mathbf{a}, \mathbf{s}')$, we consider a discrete time system by using uniformization technique [6] with a uniformization parameter u obtained as follows:

$$u = \lambda_b + \lambda_c + \lambda_s + \lambda_e. \tag{3}$$

Based on the uniformization parameter u, we can determine the probabilities of the events as follows. In the event $\mathbf{e}$, the probabilities of a request arriving from other balloons, a user, and a satellite/an aircraft are λ_b/u, λ_c/u, λ_s/u, respectively. The probability of energy arrival is λ_e/u. Then, we can derive the transition probability matrix for the system. However, to do so, we need to know the environment parameters, e.g., successful energy harvesting probability, requests' arrival rates. These parameters are generally not known in advance and building the model with complete information may not be possible. Therefore, we apply a learning algorithm based on simulation-based method [7]. The main idea of the simulation-based method is based on a "simulator" that can simulate the environment by generating environment parameters (e.g., a successful energy harvesting probability and arrival rates). Then we use the parameters from simulations to derive the admission control policy for the access point. Based on the simulation-based method, the transition probability function can be defined as follows:

$$P(\mathbf{s}, \mathbf{a}, \mathbf{s}') = p_{env} p(\mathbf{s}) p(\mathbf{a}), \tag{4}$$

where p_{env} is environment parameter (e.g., the successful energy harvesting probability), $p(\mathbf{s})$ is the probability that the system is at state $\mathbf{s}$, and $p(\mathbf{a})$ is the probability that action $\mathbf{a}$ is taken.

When there is a request $\mathbf{x}$ arriving at the access point, if the request is accepted, the balloon will receive an immediate

reward r_x corresponding to the type of the request. Otherwise, the access point gains nothing, i.e.,

$$R(\mathbf{s},\mathbf{a}) = \begin{cases} r_x, & \text{if } \mathbf{x} \in \{x_b, x_c, x_s\}, \mathbf{a} = 1 \text{ and } \mathbf{e} + \mathbf{x} \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases}$$

Note that, for the case $\mathbf{x} = x_e$, the access point will always receive energy if the battery is not full. However, there is no reward for such an action.

B. MDP with Parameterization

We consider a parameterized randomized policy [8], [9], [10]. With the parameterized randomized policy, when there is a request arriving at the access point, the request will be accepted with probability defined as follows:

$$\mu_\Theta(\mathbf{s},\mathbf{a}) = \frac{1}{1+\exp(1.5(\theta_x - \mathbf{e}))}, \tag{5}$$

where Θ is the parameter vector of the learning algorithm, $\mathbf{e}$ is the current energy level of the battery and θ_x is the parameter for requests from event $\mathbf{x}$. Additionally, the parameterized randomized policy $\mu_\Theta(\mathbf{s},\mathbf{a})$ must not be negative and meet the following condition,

$$\sum_{\mathbf{a}\in\mathcal{A}} \mu_\Theta(\mathbf{s},\mathbf{a}) = 1. \tag{6}$$

With the randomized parameterized policy, the transition probability function will be parameterized as follows:

$$P_\Theta(\mathbf{s},\mathbf{s}') = \sum_{\mathbf{a}\in\mathcal{A}} \mu_\Theta(\mathbf{s},\mathbf{a}) P(\mathbf{s},\mathbf{a},\mathbf{s}'). \tag{7}$$

Similarly, we can parameterize the immediate reward function as follows:

$$R_\Theta(\mathbf{s}) = \sum_{\mathbf{a}\in\mathcal{A}} \mu_\Theta(\mathbf{s},\mathbf{a}) R(\mathbf{s},\mathbf{a}). \tag{8}$$

We aim to maximized the average reward under randomized parameterized policy denoted by $\psi(\Theta)$ and it can be defined as follows:

$$\psi(\Theta) = \lim_{t\to\infty} \frac{1}{t} E_\Theta\left[\sum_{k=0}^{t} R_\Theta(\mathbf{s}_k)\right], \tag{9}$$

where $\mathbf{s}_k$ is the system state at step k and $E_\Theta[\cdot]$ is the expected reward of the system.

We then make some assumptions as follows:

Assumption 1. *The Markov chain corresponding to every $P \in \overline{\mathcal{P}}$ is aperiodic. Furthermore, there exists a state $\mathbf{s}^*$ which is recurrent for every of such Markov chain.*

Assumption 2. *For every state $\mathbf{s},\mathbf{s}' \in \mathcal{S}$, the functions $P_\Theta(\mathbf{s},\mathbf{s}')$ and $R_\Theta(\mathbf{s})$ are bounded, twice differentiable, and have bounded first and second derivatives.*

Assumption 1 implies that the system has a Markov property and Assumption 2 guarantees that the transition probability function and the average reward function depend smoothly on the parameter vector Θ after they are parameterized by Θ. Assumption 2 is necessary when we use the policy gradient method to adjust vector Θ. Under Assumption 2, the average reward $\psi(\Theta)$ is well defined for every Θ and does not depend on an initial state. Furthermore, we have the following balance equations:

$$\begin{aligned} \sum_{\mathbf{s}=1}^{S} \pi_\Theta(\mathbf{s}) P_\Theta(\mathbf{s},\mathbf{s}') &= \pi_\Theta(\mathbf{s}'), \\ \sum_{\mathbf{s}=1}^{S} \pi_\Theta(\mathbf{s}) &= 1, \end{aligned} \tag{10}$$

where $\pi_\Theta(\mathbf{s})$ is steady state probability at state $\mathbf{s}$ under the parameter vector, and thus the average reward function can be also defined as follows:

$$\psi(\Theta) = \sum_{\mathbf{s}} \pi_\Theta(\mathbf{s}) R_\Theta(\mathbf{s}). \tag{11}$$

C. Policy Gradient Method

We now can apply the policy gradient method [11] to update for the parameter vector Θ as follows:

$$\Theta_{k+1} = \Theta_k + \gamma_k \nabla\psi(\Theta_k) \tag{12}$$

where γ_k is a step size parameter. In the policy gradient method, we start with an initial parameter vector Θ_0, and then the parameter vector Θ will be updated iteratively based on (12). Under Assumption 2 and an appropriate step size, it was proved in [11] that, $\lim_{k\to\infty} \nabla\psi(\Theta_k) = 0$. That is, the average reward $\psi(\Theta_k)$ converges almost surely.

We now propose Proposition 1 to calculate the gradient for the average reward $\psi(\Theta)$.

Proposition 1. *Let Assumption 1 and Assumption 2 hold, then*

$$\nabla\psi(\Theta) = \sum_{\mathbf{s}\in\mathcal{S}} \pi_s(\Theta)\Big(\nabla R_\Theta(\mathbf{s}) + \sum_{\mathbf{s}'\in\mathcal{S}} \nabla P_\Theta(\mathbf{s},\mathbf{s}') \mathbf{d}_\Theta(\mathbf{s}')\Big). \tag{13}$$

$\mathbf{d}_\Theta(\mathbf{s}')$ is the differential reward at state $\mathbf{s}$ and it can be defined as follows:

$$d_\Theta(\mathbf{s}) = E_\Theta\left[\sum_{k=0}^{T-1} (R_\Theta(\mathbf{s}_k) - \psi(\Theta) | \mathbf{s_0} = \mathbf{s}\right], \tag{14}$$

where $T = \min\{k > 0 | \mathbf{s}_k = \mathbf{s}^*\}$ is the first future time that the state $\mathbf{s}^*$ is visited. Because of limited space, the proof of Proposition 1 can be found in [8].

D. Simulation-based learning algorithm

We update the parameter vector Θ iteratively based on (12) with the value of the gradient of average reward calculated from Proposition 1. However, it is not easy to calculate the terms in (13). Additionally, when the state space and action space are large, it is intractable to calculate exactly the value of the gradient of the average reward function. Therefore, in this paper, we consider the approach that can estimate the gradient of the average reward function and then the parameter vector Θ can be adjusted in an online manner.

From (6), we have $\sum_{\mathbf{a}\in\mathcal{A}} \mu_\Theta(\mathbf{s},\mathbf{a}) = 1$, so we derive $\sum_{\mathbf{a}\in\mathcal{A}} \nabla\mu_\Theta(\mathbf{s},\mathbf{a}) = 0$. From (8), we have:

$$\begin{aligned} \nabla R_\Theta(\mathbf{s}) &= \sum_{\mathbf{a}\in\mathcal{A}} \nabla\mu_\Theta(\mathbf{s},\mathbf{a}) R(\mathbf{s},\mathbf{a}) \\ &= \sum_{\mathbf{a}\in\mathcal{A}} \nabla\mu_\Theta(\mathbf{s},\mathbf{a})\big(R(\mathbf{s},\mathbf{a}) - \psi(\Theta)\big). \end{aligned} \tag{15}$$

This is from the fact that $\sum_{\mathbf{a}\in\mathcal{A}} \nabla\mu_\Theta(\mathbf{s},\mathbf{a}) = 0$. Moreover, we have

$$\sum_{\mathbf{s}'\in\mathcal{S}} \nabla P_{\mathbf{s},\mathbf{s}'}(\Theta)\mathbf{d}(\mathbf{s}',\Theta) = \sum_{\mathbf{s}'\in\mathcal{S}}\sum_{\mathbf{a}\in\mathcal{A}} \nabla\mu_\Theta(\mathbf{s},\mathbf{a})P_{\mathbf{s},\mathbf{s}'}(\mathbf{a})\mathbf{d}(\mathbf{s}',\Theta). \tag{16}$$

Therefore, along with Proposition 1, we derive the following results:

$$\begin{aligned}
\nabla\psi(\Theta) &= \sum_{\mathbf{s}\in\mathcal{S}} \pi_\Theta(\mathbf{s})\Big(\nabla R_\Theta(\mathbf{s}) + \sum_{\mathbf{s}'\in\mathcal{S}} \nabla P_\Theta(\mathbf{s},\mathbf{s}')\mathbf{d}_\Theta(\mathbf{s}')\Big) \\
&= \sum_{\mathbf{s}\in\mathcal{S}} \pi_\Theta(\mathbf{s})\Big(\sum_{\mathbf{a}\in\mathcal{A}} \nabla\mu_\Theta(\mathbf{s},\mathbf{a})(R(\mathbf{s},\mathbf{a}) - \psi(\Theta)) + \\
&\quad + \sum_{\mathbf{s}'\in\mathcal{S}}\sum_{\mathbf{a}\in\mathcal{A}} \nabla\mu_\Theta(\mathbf{s},\mathbf{a})P(\mathbf{s},\mathbf{a},\mathbf{s}')\mathbf{d}_\Theta(\mathbf{s}')\Big) \\
&= \sum_{\mathbf{s}\in\mathcal{S}} \pi_\Theta(\mathbf{s}) \sum_{\mathbf{a}\in\mathcal{A}} \nabla\mu_\Theta(\mathbf{s},\mathbf{a}) \\
&\quad \times \Big(\big(R(\mathbf{s},\mathbf{a}) - \psi(\Theta)\big) + P(\mathbf{s},\mathbf{a},\mathbf{s}')\mathbf{d}_\Theta(\mathbf{s}')\Big) \\
&= \sum_{\mathbf{s}\in\mathcal{S}}\sum_{\mathbf{a}\in\mathcal{A}} \pi_\Theta(\mathbf{s})\nabla\mu_\Theta(\mathbf{s},\mathbf{a})q_\Theta(\mathbf{s},\mathbf{a}),
\end{aligned} \tag{17}$$

where

$$\begin{aligned}
q_\Theta(\mathbf{s},\mathbf{a}) &= \Big(R(\mathbf{s},\mathbf{a}) - \psi(\Theta)\Big) + \sum_{\mathbf{s}'\in\mathcal{S}} P(\mathbf{s},\mathbf{a},\mathbf{s}')\mathbf{d}_\Theta(\mathbf{s}') \\
&= E_\Theta\left[\sum_{k=0}^{T-1} \big(R(\mathbf{s},\mathbf{a}) - \psi(\Theta)\big)|\mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right].
\end{aligned} \tag{18}$$

Here again T is the first future time that the current state $\mathbf{s}^*$ is visited. $q_\Theta(\mathbf{s},\mathbf{a})$ can be interpreted as the differential reward if action $\mathbf{a}$ is taken based on policy μ_Θ at state $\mathbf{s}$. We need to note that $d_\Theta(\mathbf{s})$ is the cost at state $\mathbf{s}$ and it is different from the different cost at state $\mathbf{s}$ under action $\mathbf{a}$, i.e., $q_\Theta(\mathbf{s},\mathbf{a})$. Then, we present Algorithm 1 to update the parameter vector Θ at the visits to the recurrent state $\mathbf{s}^*$.

Algorithm 1 Algorithm to update parameter vector Θ at visits to the recurrent state $\mathbf{s}^*$

At the time T_{m+1} of the $(m+1)$th visit to state $\mathbf{s}^*$, we update the parameter vector Θ and the estimated average reward $\widetilde{\psi}$ as follows:

$$\Theta_{m+1} = \Theta_m + \gamma_m F_m(\Theta_m, \widetilde{\psi}_m),$$

$$\widetilde{\psi}_{m+1} = \widetilde{\psi}_m + \eta\gamma_m \sum_{n=t_m}^{t_{m+1}-1} (R(\mathbf{s}_n,\mathbf{a}_n) - \widetilde{\psi}_m)$$

where

$$F_m(\Theta_m, \widetilde{\psi}_m) = \sum_{n=t_m}^{t_{m+1}-1} \widetilde{q}_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)\frac{\nabla\mu_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)}{\mu_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)},$$

$$\widetilde{q}_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n) = \sum_{k=n}^{t_{m+1}-1} (R(\mathbf{s}_k,\mathbf{a}_k) - \widetilde{\psi}_m).$$

In Algorithm 1, η is a positive scalar and γ_m is a step size parameter. We derive the following convergence result for Algorithm 1.

Proposition 2. *Let Assumption 1 and Assumption 2 hold, and let (Θ_m) be the sequence of parameter vectors generated by Algorithm 1 with a suitable step size parameter γ satisfied Assumption 3, then $\psi(\Theta_m)$ converges and*

$$\lim_{m\to\infty} \nabla\psi(\Theta_m) = 0,$$

with probability 1.

The proof of the Proposition 2 can be found in [8].

Assumption 3. *The step size γ_m is deterministic, nonnegative and satisfies the following condition,*

$$\sum_{m=1}^{\infty} \gamma_m = \infty, \quad \sum_{m=1}^{\infty} \gamma_m^2 < \infty.$$

In Algorithm 1, to update the value of the parameter vector Θ at the next visit time to the state $\mathbf{s}^*$, we need to store all values of $\widetilde{q}_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)$ and $\frac{\nabla\mu_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)}{\mu_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)}$ between two successive visits. However, this method could result in slow processing. Therefore, we modify Algorithm 1 to improve the efficiency. First, we rewrite $F_m(\Theta_m, \widetilde{\psi}_m)$ as follows:

$$\begin{aligned}
F_m(\Theta_m, \widetilde{\psi}_m) &= \sum_{n=t_m}^{t_{m+1}-1} \widetilde{q}_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)\frac{\nabla\mu_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)}{\mu_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)} \\
&= \sum_{n=t_m}^{t_{m+1}-1} \frac{\nabla\mu_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)}{\mu_{\Theta_m}(\mathbf{s}_n,\mathbf{a}_n)} \sum_{k=n}^{t_{m+1}-1} \big(R(\mathbf{s}_k,\mathbf{a}_k) - \widetilde{\psi}_m\big) \\
&= \sum_{n=t_m}^{t_{m+1}-1} \big(R(\mathbf{s}_k,\mathbf{a}_k) - \widetilde{\psi}_m\big) z_{k+1},
\end{aligned} \tag{19}$$

where

$$z_{k+1} = \begin{cases} \frac{\nabla\mu_{\Theta_m}(\mathbf{s}_k,\mathbf{a}_k)}{\mu_{\Theta_m}(\mathbf{s}_k,\mathbf{a}_k)}, & \text{if } k = t_m, \\ z_k + \frac{\nabla\mu_{\Theta_m}(\mathbf{s}_k,\mathbf{a}_k)}{\mu_{\Theta_m}(\mathbf{s}_k,\mathbf{a}_k)}, & k = t_m+1,\ldots,t_{m+1}-1. \end{cases}$$

The detail of the algorithm can be expressed as in Algorithm 2, where η is a positive scalar and γ_k is the step size of the algorithm.

Algorithm 2 Algorithm to update Θ at every time step

At a typical time k, the state is $\mathbf{s}_k$, and the values of Θ_k, z_k, and $\widetilde{\psi}(\Theta_k)$ are available from the previous iteration. We update Θ and $\widetilde{\psi}$ according to:

$$z_{k+1} = \begin{cases} \frac{\nabla\mu_{\Theta_k}(\mathbf{s}_k,\mathbf{a}_k)}{\mu_{\Theta_k}(\mathbf{s}_k,\mathbf{a}_k)}, & \text{if } \mathbf{s}_k = \mathbf{s}* \\ z_k + \frac{\nabla\mu_{\Theta_k}(\mathbf{s}_k,\mathbf{a}_k)}{\mu_{\Theta_k}(\mathbf{s}_k,\mathbf{a}_k)}, & \text{otherwise,} \end{cases}$$

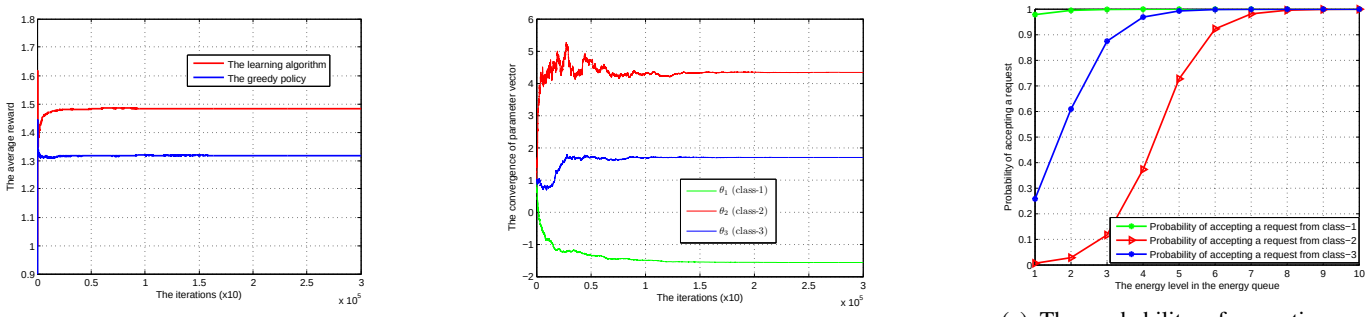
$$\Theta_{k+1} = \Theta_k + \gamma_k(R(\mathbf{s}_k,\mathbf{a}_k) - \widetilde{\psi}_k)z_{k+1},$$

$$\widetilde{\psi}_{k+1} = \widetilde{\psi}_k + \eta\gamma_k(R(\mathbf{s}_k,\mathbf{a}_k) - \widetilde{\psi}_k).$$

IV. Numerical Results

A. Experiment Setup

In this section, we perform simulations using MATLAB to evaluate the performance of the proposed learning algorithm. In the experiment, we consider the scenario as depicted in



(a) The average reward

(b) The parameter vector

(c) The probability of accepting requests from different classes of user

Fig. 3: The convergence and the policy of the learning algorithm.

Fig. 2. The maximum queue size is set at 10 units. There are three classes of users, namely, class-1, class-2, and class-3, corresponding to requests from balloons, users on the ground, and satellites/aircrafts, respectively. The arrival rates of requests from users of class-1, class-2, and class-3, are 60, 70, and 10 requests per hour, respectively. When a request is accepted, the access point will use one unit of energy from the battery to serve for the request (i.e., to receive data and transmit the data to the destination). Upon accepting requests, the access point receives the rewards of 5, 2, and 3 monetary units for class-1, class-2, and class-3, respectively. The energy arrival rate is 110 per hour and the successful energy harvesting probability is 0.9. If the balloon harvest energy successfully and the battery is not full, the battery will increase one unit. For the learning algorithm, the initial parameter vector is set at $\Theta = (\theta_1, \theta_2, \theta_3) = (1, 1, 1)$, and the chosen initial estimated average reward is 0.7.

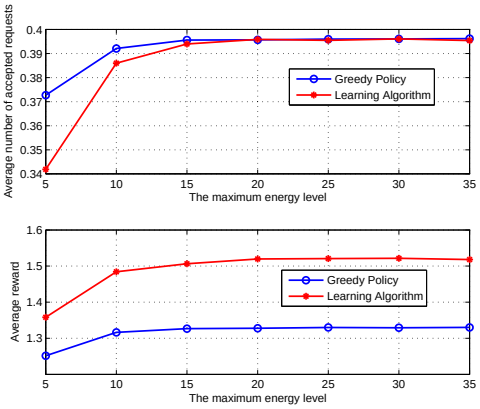


Fig. 4: The performance of the system when the maximum queue size is varied.

B. Numerical Results

We first consider the convergence of the proposed learning algorithm (i.e., Algorithm 2). Figures 3a and 3b show the convergence in the terms of the average reward and the parameter vector. In both figures, the proposed learning algorithm converges within around 5.10^5-10^6 iterations. In Fig. 3a, we also compare the average rewards obtained by the learning algorithm and the greedy policy that always accepts requests. At the convergence points, the average reward obtained by the learning algorithm reaches approximately 1.48 which 8.8% higher than that obtained by the greedy policy.

In Fig. 3b, the parameter vector Θ converges to (-1.5577, 4.3448, 1.7029) for class-1, class-2, and class-3, respectively. Then, from the parameter vector obtained from the learning algorithm, we can determine the policy for the access point as shown in Fig. 3c. In Fig. 3c, the requests from other balloons will be always accepted, while the requests from users on the ground and satellites will only be accepted only when the energy level in the battery is high enough. In particular, the access point will accept the requests from a user on the ground and satellites when the energy level is higher than 5 units and 2 units, respectively.

We then evaluate the impacts of the battery capacity to the performance of the system. Specifically, in Fig. 4, we vary the maximum battery capacity and observe its impacts to the average number of accepted requests and the average reward of the access point. When the maximum battery capacity increases, the average reward and average number of accepted requests obtained by the learning algorithm (LA) and the greedy policy (GP) will increase and saturate when the maximum battery capacity is greater than 15. However, it is interesting that when the maximum battery capacity increases from 5 to 15, the average number requests accepted by the learning algorithm is lower than that of the greedy policy. However, the average reward obtained by the learning algorithm is higher than that of the greedy policy. The reason can be found from the policy of the learning algorithm and the policy of the greedy policy as shown in Fig. 5. While the greedy policy always accepts requests if the battery is not empty, the learning algorithm selectively accepts requests from class-2 and class-3 when the energy level is high enough. It is also worth to note that, when the maximum battery capacity is greater than 15, the average numbers of requests obtained by the learning algorithm and the greedy policy are equal. However, the average reward obtained by the learning algorithm is always greater than that of the greedy policy. The reason is because when the battery capacity is small, the amount of energy harvested will be limited and thus learning algorithm will accept requests which yield high reward and reject requests which yield low reward. When the battery capacity increases, the amount of energy harvested will increase, and thus, there is more chance for the requests with low reward to be accepted (as shown in Fig. 5).

However, when the battery capacity is greater than 15, the performance of the system will be saturated. The reason is, the number of accepted requests depends not only the battery capacity, but also on the energy arrival rate. In other words, the system performance is constrained by energy arrival, if the battery capacity is large enough.

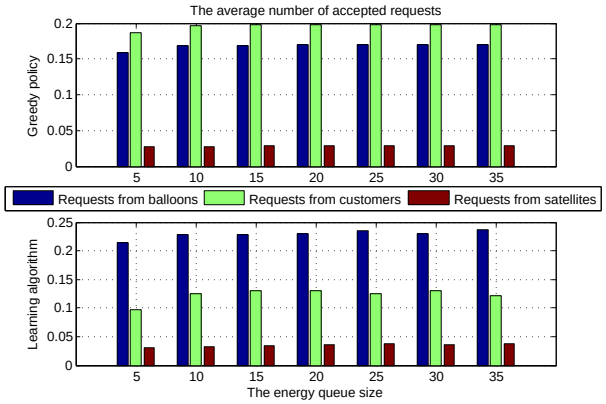
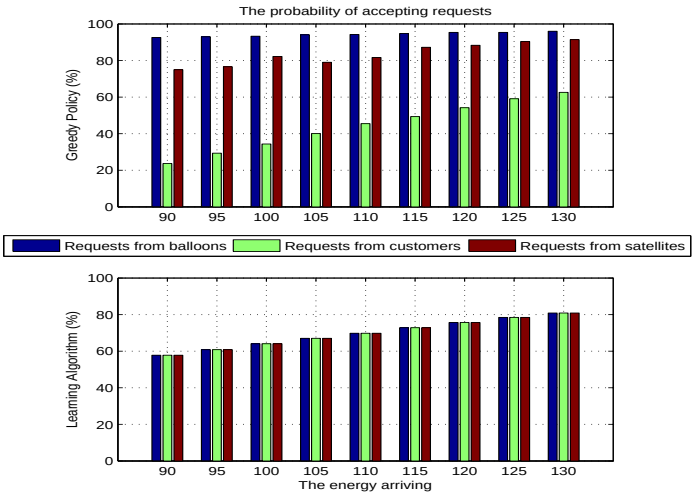


Fig. 5: The average number of accepted request.

We fix the battery capacity at 10 and vary the energy arrival rate. When the energy arrival rate increases from 90 to 130, the probability of accepting requests by the greedy policy and the learning algorithm will increase. As a result, the average reward as well as the average energy harvested by both policies will also increase. Moreover, as shown in Fig. 6, when the energy arrival rate is small, the average reward and the average energy in the battery with the learning algorithm are much greater than those of the greedy policy. However, when the energy arrival rate increases, the performance gap between the learning algorithm and greedy algorithm becomes smaller. Eventually, when the energy arrival rate is large, the results obtained by the greedy policy will approach those of the learning algorithm.



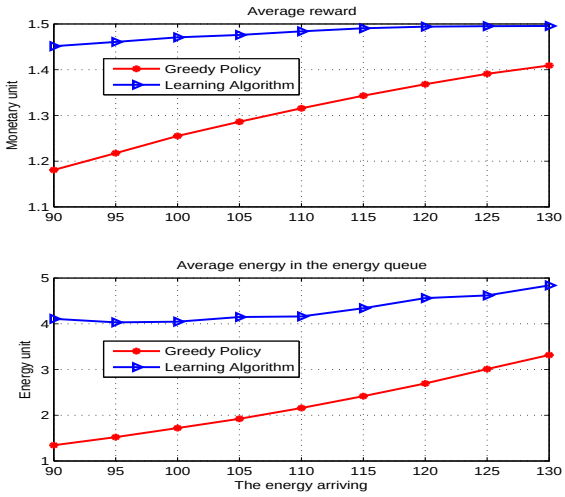


Fig. 6: The performance of the system when the energy arrival probability is varied.

V. Summary

In this paper, we have studied and developed an optimization model for the optimal energy control problem for a network in the sky. The aim is to maximize the network performance as well as the profit for the network provider. We have first formulated the problem as a Markov decision process and then applied an online learning algorithm based on the gradient method to obtain the optimal policy for the access point deployed on a balloon. The numerical results have been presented to show the impacts of parameters to the system performance as well as to show the convergence and the efficiency of the proposed learning algorithm.

References

[1] Project Loon - Google, http://www.google.com/loon/

[2] "Percentage of Individuals using the Internet 2000-2012", International Telecommunications Union (Geneva), retrieved 22 June 2013.

[3] Y. Hu, and V. O. K. Li, "Satellite-based Internet: A tutorial," in *IEEE Communications Magazine*, vol. 39, issues 3, pp. 154-162, March 2001.

[4] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," in *Fourth International Symposium on Information Processing in Sensor Networks*, pp. 457-462, April 2005.

[5] A. Georgiadis, G. Andia, and A. Collado, "Rectenna design and optimization using reciprocity theory and harmonic balance analysis for electromagnetic (EM) energy harvesting," in *IEEE Antennas and Wireless Propagation Letters*, vol. 9, pp. 444-446, 2010.

[6] R. G. Gallager, *Discrete stochastic processes*, Kluwer Academic Publishers, London, 1995.

[7] Abhijit Gosavi, "Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning," Springer Press, 2003.

[8] P. Marbach, and J. N. Tsitsiklis, "Simulation-based optimization of Markov reward processes," in *IEEE Transactions on Automatic Control*, vol. 46, pp. 191-209, Feb. 2001.

[9] O. Buffet, A. Dutech, and F. Charpillet, "Shaping multi-agent systems with gradient reinforcement learning," *Journal of Autonomous Agents and Multi-Agent System*, vol. 15, pp. 197-220, Jan. 2007.

[10] J. Baxter, P. L. Barlett, L. Weaver, "Experiments with infinite-horizon, policy-gradient estimation," *Journal of Artificial Intelligence Research*, vol. 15, pp. 351-381, Nov. 2001.

[11] D. P. Bertsekas, *Nonlinear programming*, Athena Scientific, Belmont, MA, 1995.